8-6-2011

Mining Transaction Data for Process Instance Monitoring in Legacy Systems

Jyoti Bhat
Infosys Technologies Limited, jyotimb@infosys.com

Sukriti Goel
Infosys Technologies Limited, sukriti_goel@infosys.com

Mining Transaction Data for Process Instance Monitoring in Legacy Systems

Jyoti M. Bhat
Infosys Labs, Bangalore
JYOTIMB@infosys.com

Sukriti Goel
Infosys Labs, Bangalore
IITB Monash Research Academy, Mumbai
sukriti_goel@infosys.com

ABSTRACT (REQUIRED)

End-to-End business processes in organizations are implemented across multiple applications, legacy systems, ERP systems and products. In such scenarios where applications are developed over a period of time and with varying technologies, monitoring end-to-end business processes is a challenge. Typical methods for providing process monitoring capabilities are intrusive methods like changing code and introducing probes; or introducing new software tools like EAI and BAM. We propose a non-intrusive process instance monitoring (PIM) method that uses the persistent data generated by the business transactions to monitor the process instances in Legacy Information Systems. We propose a slightly unconventional data mining method where the transaction data is parsed from the application data stores, loaded into custom schema and then associated to the process flow for monitoring the state of individual process instances. The approach further provides for alerting when business events like an SLA violation occur.

Keywords (Required)

Data Mining, Process Monitoring, Legacy Information Systems.

INTRODUCTION

In the globalized competitive market, organizations need greater visibility into the process execution to maintain control over their business operations. Managers require monitoring tools that help in identifying exception situations and facilitating timely decision making to ensure adherence of service level agreements (SLA) and compliance requirements. Operational control requires visibility into the process execution at a process instance level. Individual applications and departments implement various means of monitoring system and functional level parameters. Most large organizations have their core business processes implemented across many applications, legacy systems, ERP systems and products. The processing logic and rules are coded in algorithms, batch jobs, database constraints etc. The process flow in such scenarios is achieved by the integration and data hand offs between applications, systems and programs and hence is not explicit. A single end to end business process is implemented across more than one application with many of them being process unaware systems (Leymann, Reisig, Thatte and van der Aalst 2006). In such situations providing process instance-level visibility for the end-to-end business processes would involve expensive IT system changes to get the required process information.

Sophisticated tools and methods are available for monitoring at the level of data and IT systems but providing business process level monitoring capabilities in legacy information systems has limited rudimentary approaches. Usually changes are made to the application code to identify exception events and send an alert. This is an expensive approach as the application code has to be modified, tested and redeployed. Another approach involves adding probes in the middleware or adding database triggers to monitor each process instance at run-time. Adding probes can hamper system performance, and is risky as it involves changes, even if minor, in a production environment as well as significant effort in testing the applications. Adding database triggers similarly deteriorates the performance of the existing process. Technologies like Business Process Management Systems (BPMS) with its out-of-the-box process monitoring feature and Business Activity Monitoring (BAM) tools can be implemented to provide the process monitoring capabilities on legacy systems. But such implementations involve investments in process and application level changes to ensure that the process execution data at the instance level is available to the BPMS/BAM layer in the required format. There are other people related challenges which make the above approaches unsuitable like lack of skilled programmers or knowledge in the technology of the existing legacy systems; Fear of introducing errors into the complex application which currently works fine; lack of consensus between the application business owner and the business unit which requires visibility for investments in process and system changes. In such situations there is a need to explore other methods for providing process visibility without having to make any changes or intrude into the execution of the processes implemented in the existing systems.

We propose Process Instance Monitoring (PIM), a non-intrusive method of providing process instance-level monitoring in systems by using the data generated and persistent in the information systems. When a business process executes in the IT

systems the process instance logs transaction information in various data sources like database tables, files, logs, message queues, etc. PIM provides visibility at the process instance-level execution by reading the state of the transaction data and inferring the status of each process instance and associating it to the process flow available in the process models. This requires each process instance and the related data to be uniquely identifiable and associated to a process definition model. The method is non intrusive as it does not intrude into the execution of the business process and does not add probes into the execution layer of the IT systems (like the middleware, network or database). The method does not require any changes to be made to the current systems executing the process. The approach further provides for alerting when business events like an SLA violation occurs which enables the process user to take alternate preventive/corrective action for the specific business process instance.

In the next section the challenges involved in Process Instance Monitoring in Legacy Information Systems are described. The subsequent sections provide an overview and details of the PIM method. We provide a case study to illustrate how the PIM method was applied to meet the process monitoring requirements. We discuss and compare other related work and monitoring technologies with our approach. In the end we discuss the conclusion and future work in this area.

CHALLENGES

Flow of the Process to be monitored is not known: The objective is to monitor the process instances in process-unaware legacy information systems. As the process model is not explicit in legacy systems, the process flow to be monitored is not available. The first challenge is to discover the as-is process model. This challenge however can be addressed by using process elicitation methods (Verner 2004; Basili and Weiss 1984; Wolf and Rosenblum 1993) or process mining techniques (van der Aalst, Weijters and Maruster 2004) to discover the process and create the process models. Once the as-is process is known, the next challenge is to simplify the process model and identify the process flow which is to be monitored. This may involve creating a monitored process flow by removing activities (i.e., not displaying the activities in the model) which are required to be monitored from the user point of view.

Missing process state information: Process Aware Information systems (PAISs) like BPMS, workflow systems, ERP, etc., have explicit process definitions and execute the process as sequences of activities and maintain the status of execution (Dumas, van der Aalst and Ter Hofstede 2005). The concept of a business activity is missing in traditional or legacy information systems as the process flow is implicit and implemented by invoking methods, procedures and programs (callable units). While the process flow can be documented, the challenge of not knowing the state of business activity instances and process instances still exists. The process state information can be obtained from legacy systems by making code changes, using probes and instrumentation of code. But in practice this is not possible as stakeholders do not allow code changes in the production environment. Insufficient information and skills regarding the code and programming language as well as unavailability of source code (e.g., third party product) are some additional challenges faced by organizations.

Manual activities: The business process may contain manual activities which usually have a higher probability of delay as they are dependent on availability of people and other resources. Monitoring manual activities is a challenge using the automated tools thus we would ignore this challenge while discussing PIM.

Identifying the data sources: An end-to-end business process spans multiple applications and technologies. Identifying the relevant systems and the format of the data sources of these systems is a challenge. If any of the data sources from the systems is missed, the results of monitoring will be erroneous.

Identify data traces for each activity: To monitor the state of a business activity, the exact data trace or data event which indicates the state change of the business activity needs to be identified. For example, in an order to remittance process the *create order* activity may be completed when a new row is inserted in *order_details* table of order entry system, while the activity *order ready for shipping* is completed when an entry is made in flat file generated by a batch process in a mainframe based order provisioning system. A single business activity instance may leave multiple data traces as it will update various data sources during execution. The challenge is to identify the most appropriate data trace from the multiple data traces. In some cases the data trace created by business activities is overwritten by other subsequent business activities.

Correlation of business activities: As legacy systems do not maintain a process identifier while storing process execution data, the business activity instances need to be correlated to the correct process instances. There may be several process instances executing simultaneously and hence correlation of business activity instances to process instances becomes a challenge.

Unavailability of adaptor: PIM needs to connect to various data sources thus the adaptors should be available for all data sources. In case the adaptor is not available the challenge is to create a new adaptor so that it can work with other components of PIM seamlessly.

Real time monitoring capability: The process instance monitoring is useful from operations management point of view and to provide the relevant alerts to the concerned people. If the approach for process monitoring is non-intrusive, the process monitoring cannot be real-time. It is a challenge to achieve the process monitoring as near-real time as possible.

Process Instance Monitoring Approach is explained in detail in the next few sections.

PROCESS INSTANCE MONITORING (PIM) OVERVIEW

Non-intrusive process instance monitoring implies that the business process instances can be monitored across systems independent of the technology and platform, as it is not necessary to modify code or introduce probes into the middleware. For a given business process, the process instance execution is traced across systems using the persisted data by identifying events of relevance to monitor the process instance progress. The PIM approach relies on the premise that every business activity which is executed as part of a business process leaves its footprint in the form of data traces such as transaction data or audit logs in databases, web logs, application server logs or flat files. PIM makes use of these data traces to capture the state of process instances. The persistent data is read in a non-intrusive manner using adaptors.

As the process instances are executed, PIM identifies pre-defined events like missing an SLA and raises alerts thereby leading to improved exception handling and SLA monitoring. For a specific event, the user can be alerted to take an alternate action or the system can be configured to trigger an alternate action. PIM consists of the following high level capabilities:

(1) Identifying the creation of new process instances;

(2) Tracking the status of each process instance against the process flow in near real time;

(3) Raising alerts for each transaction as per pre-defined rules;

(4) Identifying the process activity which caused the alert.

To provide these capabilities PIM needs the following:

Awareness of the flow of activities of the process(es) being monitored: The processes are captured in processes models by studying existing operations and system documentation. The as-is process model can be mined using the process mining techniques [16, 17, 18] or by using process elicitation methods [6] in case the process is not known. The as-is process model is then converted into a monitored process model by simplifying it. Activities which need to be monitored and need an alert in case of delays or which trigger an alternate action or process are usually part of the monitored process model.

Ability to uniquely identify each process instance in the system: PIM assigns an identifier to each process instance and business activity instance. This is used to correlate the business activity to the process instance. The exact value of the process instance identifier can change as the process state changes based on the business activities completed. For example, if the value of order id is assigned as the process instance identifier, it can change to a combination of order id, invoice id, shipping id etc as the process instance executes. The process instance identifier or correlation identifier is progressively built using the activity identifiers based on the activities which match the existing process correlation identifier. If the activity has more than one identifier (i.e., a composite identifier), all the correlation identifiers are added to the process sequence identifier.

Ability to infer the execution (completion) of process activities from the business transaction data: The updates or changes to the business transaction data due to process activities are parsed, converted into events and processed to identify the execution and completion of process activities and hence the associated process instance. For example, Insertion of a row in order table can be used to infer that the Create Order activity is complete.

Ability to capture the current state of individual process instance: As the process instance progresses, the PIM captures and stores the state of the instance which includes data related to various process metrics like queue time, processing time and other process specific data. The data about the current status of process instances and business activities can be used for advanced process analysis as well.

Raise alerts when defined transactions violate pre-defined rules: Each process instance initiated is verified against the pre-defined rules and alerts are raised accordingly. The alerts can be leading alerts and can be triggered before the pre-defined rule is violated or can be lagging alerts and are triggered after the pre-defined rule is violated.

Figure 1 shows the different components of a proposed system based on PIM. Process Repository stores the monitored process models that contain the activities and the flow of the processes. The pre-defined rules against which the process execution is tracked namely the KPIs and SLAs are stored in the KPI definition database. Event handler is the run-time component which consists of multiple adaptors with capability to read different data sources. These adaptors poll the data

sources and identify the new data traces created since last poll. The event handler then consolidates data traces from each data source and creates events. The events can be of type *process start, process end, activity start* and *activity end*. Each event is assigned a unique correlation id as pre-defined. Each event is correlated to the relevant process model and process instance using the correlation id by the event-process correlator. Process instances are created when a *process start* event is encountered and a process instance is progressed to the next step when an *activity start* or *activity end* event is processed. The current process instance details are updated into the transaction database. The event-process correlator maintains a queue of process instances and the subsequent activities to be performed for each incomplete process instance. The alerts engine maps the process monitoring requirements defined in the KPI definition database against the status of the various process instances stored in the transaction database and raises alerts. A dashboard component delivers the status of each business transaction and flags the alerts as monitored by the PIM.

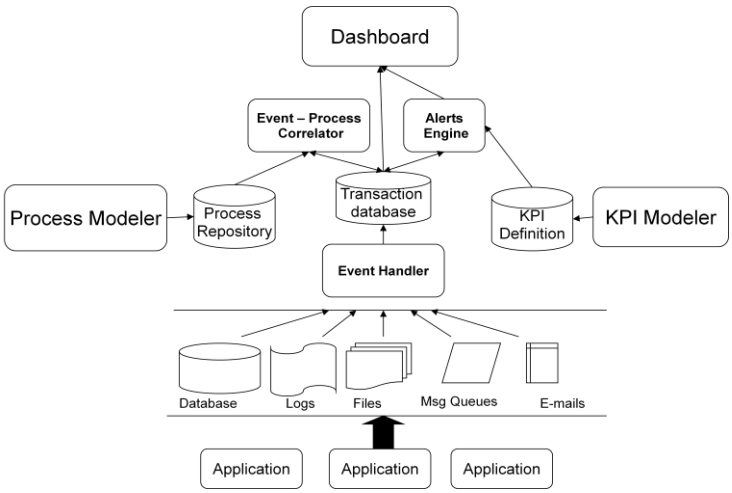

Figure 1. PIM Architecture

DETAILED APPROACH

Process Instance Monitoring system requires information on the business process, business rules against which the process is monitored, requirements for alerts, details of the data traces created by information systems to be provided to implement the monitoring capability. PIM has the following three-stage flow for implementing non-intrusive monitoring.

Setup Activities

The business processes to be monitored are modelled using standard process modelling tools. The process model is simplified to capture only those activities which need to be monitored.

Once the process model to be monitored is identified, the different systems where the business activities execute are identified with the help of the IT group. Data sources of the multiple like database management systems, flat files, message queues, log files, etc, are studied to identify candidate data events. The event handler of PIM connects with data sources using available adaptors.

The events to be identified are *process start and process end* for each process and *activity start* and *activity end* of each activity in all the processes. The event *activity end* helps to track the process instance while the other events capture improvised data which can be used for accurate analysis. Hence the *activity end* event is mandatory while the others are optional. Identifying the right data trace for each event is important as there can be multiple data traces available for a single event and for some events data trace is not available or is overwritten. The data trace for each event should have a unique identifier which can distinguish one trace to another. For example, in case of *activity end* event of *create order* the data field *order id* is the unique identifier. The data trace should also have the timestamp information of when the data trace was created. For e.g., the data field *order creation date* would be the timestamp for the *create order* data trace.

The data trace which does not get overwritten at later point of time and is updated more frequently and is easy to read and decipher should be used. For example it is better to choose the data trace in an audit table over data trace in the transaction table if there is a possibility of the transaction table being over-written. Also it is better to choose a transaction table over flat files which are written by batch programs and are available for processing only at the completion of batch program. From a different context the data trace in an RDBMS is better than that in a log file as it is easier for the user to configure the monitored process for data trace RDBMS as compared to log file.

We assume that the data trace is created after the business activity is completed as we find that it is easy to identify the *activity end* event. In case *activity end* event is not available for an activity (if activity is manual or data is overwritten) in such case the activity cannot be monitored. The setup activities involve creating the process model to be monitored, identifying and configuring the data events to the activities and identifying the process correlation ids and their values.

Tracking the process execution

The algorithm for tracking the process execution by mapping the data events to the process instances is shown in Figure 2. As the processes execute across the systems the data from identified data sources is extracted and transformed at pre-defined polling interval. The data change events are collected from multiple sources and mapped to events such as *process start, activity start, activity end* and *process end* as per definition. The events created relate to different processes and processes instances. Each event is associated to exactly one process instance and the current state of process is updated accordingly. Each *process start* event triggers a new process instance. Once the process instance is created, the subsequent activities in the process are correlated to a process instance using the correlation id. The events created from the execution data contain relevant information to correlate the event to the right process instance.

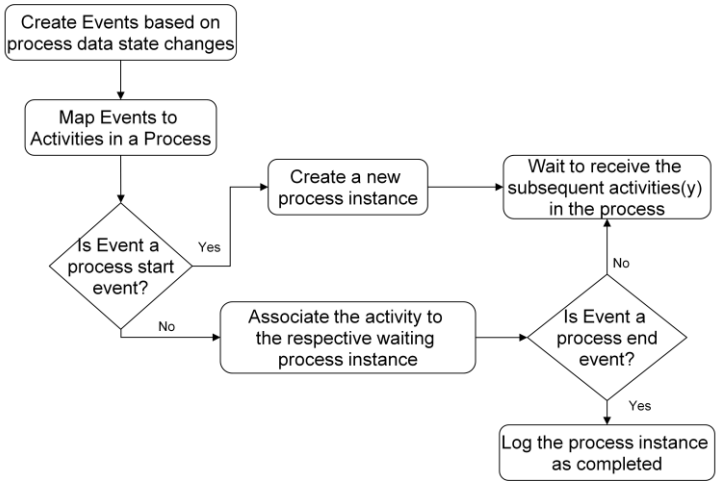

Figure 1. Algorithm to map process data events to process instance

The core of tracking the progress of a process instance is to correlate an activity to the right process instance. For every process instance created, the subsequent activity/activities in the flow are inferred using the process model. The correlation identifier of the subsequent activities is known as well. At any given point of time all active process instances and subsequent activities for those process instances are known. When an event is processed, it is correlated with the correct process instance by matching the correlation identifier of the event and the subsequent activities of the active process instances. The process instance and activity instance is updated in the transaction database. The next subsequent activity for the process instance is inferred and the process instance waits for the matching event to be processed. In case the completed activity was the last activity of the process and there are no more subsequent activities, the process instance is considered completed and updated in transaction database accordingly.

The subsequent activity in a process execution flow depends on the various paths that the process can take based on the data. These paths are available in the process models and the subsequent waiting activity for the process model can be inferred. The logic to infer the subsequent activity for workflow pattern Fork is illustrated below as an example.

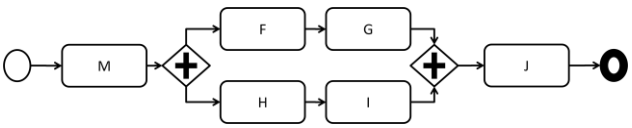

Figure 3. Process with Fork construct

For a Parallel-Split (AND-split) or Inclusive Gateway, where more than one path is possible, there is more than one activity which is tagged as the subsequent for the process instance. When one of the subsequent activities is completed, the remaining subsequent activities are maintained till completed. For e.g. in Figure 3 once F is completed, H is still maintained as subsequent activity of M.

Similar inferring logic can be developed for all other workflow modelling constructs like sequence, exclusive-or, loop, join etc.

Completed Activity	Subsequent Waiting Activities
M	F, H
F	G, H
H	G, I
G	J, I
I	J

Table 1: Subsequent Activities for a Fork construct

Monitoring Process Instances

As the process instances are tracked along the process model the data of the process instance progress is transformed and loaded into custom schema. This data about process instance progress can be verified against pre-defined KPIs and SLAs. The specific activity in a process flow which caused the KPI/SLA violation for a process instance can be flagged while monitoring.

The process monitoring data extracted, transformed and loaded by PIM in transaction database can be used later for mining business process intelligence reports, performing analysis for process performance and auditing. This data can be provided to BAM tools for advanced analysis. Thus PIM can complement BAM tools in scenarios where legacy systems cannot be changed or instrumented to collect the data required for analysis.

CASE STUDY

We use the case study of a telecom major in Asia-Pacific region where the proposed approach was used for monitoring process instances. We highlight the challenges faced and how it was addressed using the PIM approach.

Background: The order entry system of a large Telecom organization is a GUI based front end system and the processing of orders is handled by multiple systems on different technologies like web based, client server and legacy applications on Mainframes. The requirement was to monitor the hand-off of orders from the internet based order-entry front-end system to downstream systems and back. The objective was to ensure that no orders got dropped off or missed during the hand-offs which could cause delays in the order processing.

Process details: The order details are entered by the Sales Representative into the internet based system, the order is then processed and completed which involves multiple steps in order provisioning system. The order provisioning system is a legacy system and has an in-built capability to monitor the status of an order. The details of each order created are transferred to the order provisioning system using a custom built data integration application. For each order entered in order entry system, a new request file with the order details is created and the file is placed in a specific folder of the front end system. The intermediate data integration application picks up the file and enters the order details in the order provisioning system. Once the order is created or if there is an error reported by the downstream system, the response details of each order is captured as a file and the file is placed in the 'Response' folder in the front end system. The intermediate application polls the Response folder and for each new response it notifies the user about the status of the order. The order once created in the downstream system is picked up by various users and systems for further processing. But if a request file is not created due to an error in the data integration application or in the order entry system, the order is not translated into the legacy system. In such cases, the order is lost as the business user also does not get any notification of the error.

Business Challenges: A dropped or missing order from the order entry system can be identified only when a customer complaint is received or when a customer query comes. The Sales Representative then had to follow up with the IT production support team to track the order as there is no mechanism to monitor the order status before it enters the legacy system. As the information on the lost orders was not readily available in the system, the production support team had to analyse the data for the previous few days to identify the missing order and reinitiate the process of sending it to legacy system. This led to delays in order processing and customer dissatisfaction due to dropped orders and missed SLAs. Identifying the lost order and fixing it involved considerable effort and time for the production support team.

Technical Challenges: The order entry process is implemented by data hand-offs across the systems. Hence when an order is lost due to a problem at the system level, there is no log of the lost order. The production support team had to go through the orders and find out the details of the lost order manually. Production support work was outsourced to another organization so the telecom organization did not allow them to make changes in the existing code to enable proactive fault finding.

Solution: PIM was implemented to track each order created at the front end. Though the process of order provisioning is quite large, we did not have to monitor all of it. The scope of this case was to monitor the hand-off between the order entry system and order provisioning system so we simplified the process to 2 steps. One step of sending the order details from order entry system to order provisioning system and another step of receiving the response back from order provisioning system. We decided to use the first step as *process start* event. Thus in this scenario we had only one step process which needed to be monitored as shown in Figure 4.

Next step was to identify the data traces created by *process start* event, *activity start* and *activity end* event. We mapped the creation of the request file as *process start* event and receiving the response file as *activity end* event for activity *Receive Response*. We could not find the *activity start* event but it was not a critical event for tracking the process. Hence it was not possible to monitor the queue time of the activity as the *activity start event* was not available for calculations. In this case we assumed that *process start* timestamp is the timestamp of *activity start* event of *Receive Response* activity.

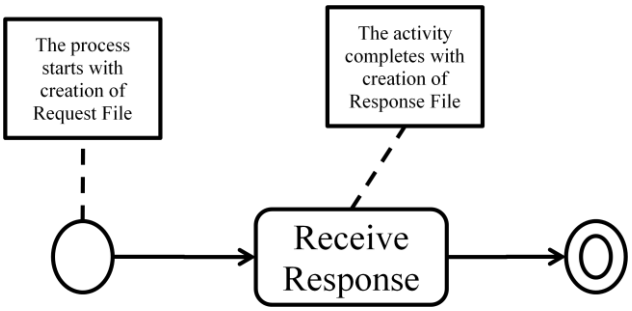

Figure 2. Process Model

The information required to correlate the two events was available in the file names. The file name contains the order id to uniquely identify an order. The format of the request file name was REQ_<order_id>.dat and format of response file name was RES_<order id>.dat. We set the correlation identifier of the activity *Receive Response* as substring of filename containing order id.

Since the maximum order processing time was 45 minutes, the SLA for the process was set for 1 hour to avoid any incorrect leading alerts. An alert could be sent by PIM to the production support team when a response for an order was not received within an hour. The polling frequency for capturing the events was set to an hour.

When a new request file is created, it indicates a *process start* event, and the creation of new response file indicates the *activity end* of *Receive Response* activity. The events are polled periodically (1 hour in this case) and then processed, thus creating the data for process monitoring. Any process instance waiting for a response file for more than an hour is considered a lost order and an alert is sent to the production support team with details of the order. The production support team is able to identify the orders immediately and manually reinitiate the order processing well ahead of the business user raising a ticket. This eliminates the delays in order processing and ensures timely processing of orders.

The adaptor to identify file creation as a data event was developed in 5 days. The setup activities such as creating the process model, configuring the events and SLA were completed in 5 days. The process models and identification of problem areas and data sources or traces was done along with the IT group in a workshop of 3 hour duration. Thus the entire effort for creating a process monitoring solution took around 2 weeks of effort.

The alternate options or solutions to achieve the above objective were either modifying the front end GUI system or implementating a system monitoring tool. In both the alternatives the estimated IT effort to implement the solution was greater than 4 weeks. The cost involved in modifying and testing the front end system or in procuring a monitoring tool was also considerably greater than cost of implementing the PIM approach. Here we have discussed a case where the process model has only one problem area. But we have found during our subsequent implementations that the effort and cost involved for implementing the PIM approach increases only marginally (about 20%) even when there are 10 problem areas to monitor. This is because effort is involved only in customizing the adaptors for the new data traces. But for modifying the

existing systems, each problem area to be monitored would increase the IT effort by at least 80% as each of them have to be developed and tested.

By implementing the PIM approach, the case study organization was able to identify and reinitiate dropped orders within an hour of the error, which reduced the customer queries on order delays due to the data hand-off errors to zero. It also reduced effort of the production support team as it re moved the need to analyse and locate the lost orders. There were other business benefits due to the reduction in the dropped orders like increased customer satisfaction and greater adherence to order processing SLAs, but we did not collect and analyse the impact on these parameters.

We chose the above case study with just 2 monitoring steps to showcase the PIM approach. The case study highlights the fact that all the process steps do not have to be tracked to monitor an end-to-end business process. We can identify the problem areas and hand-off points and monitor only some steps of the process. The case study showcases how process instance-level monitoring can be achieved by mining the transaction data and correlating the data trace to the business activities.

RELATED TECHNOLOGIES

As information technology systems are the major enablers and components of business processes there is a need for providing detailed, real time and accurate information of the business process being executed within the IT systems. The visibility into the process execution is provided through reports, dashboards, alerts, etc. While process monitoring requirements are a part of the requirements when a system is being developed, the continuous changes to the systems due to process and technology changes create a disconnect between the monitoring requirements built into the system and those required by the business users. This problem is more prominent in legacy systems than in the newer generations' applications built using process aware information systems like application servers, workflow systems, Business Process Management Systems (BPMS). While a certain amount of process level visibility is available through the process aware systems, process monitoring technologies with enhanced capabilities are being developed. These process monitoring approaches cannot be easily implemented on to process unaware legacy systems without intruding into the system making them costly and risky.

Business activity monitoring (BAM) tools provide real-time process information by accessing data from multiple application systems and other internal and external sources (Luckham 2004). BAM can be implemented in systems that use some integration mechanism like Enterprise Application Integration (EAI), Service Oriented Architecture (SOA) or BPMS. In legacy systems BAM implementations need some probes in the code or data layer for getting real-time information.

While BAM provides real-time process information, Business Intelligence (BI) uses historic process information and provides post-facto analysis to improve the business operations (Jourdan, Rainer and Marshall 2008). BI cannot be used for real time process visibility as it works off process data stored in the data warehouse. Business process intelligence (BPI) (Grigori, Casati, Catellanos, Dayal, Sayal and Shan 2008) provides both BAM and BI capabilities on top of the BPMS by applying data warehousing and data mining techniques to business process execution data. BAM, BI and BPI process simple events. Complex Event Processing (CEP) technology processes events from multiple sources across the IT layers and helps detect the complex situations (Luckham 2002). Another related event processing technology is the event stream processor which provides visibility into business operations from streaming events emitted from systems, including hardware systems Cohen, Ramesh and Chen 2005; Abadi, Ahmad, Balazinska et al 2005; McGregor and Schiefer 2004) . While CEP technologies claim to take care of event processing within a process and provide process execution information (Ammon, Springer and Wolff 2008; Kharbili 2008; Sen 2008) investment into CEP can be an overkill for processing simple events which are required for operations control.

Process Mining (van der Aalst et al 2004) uses event logs from transactional information systems like ERP, CRM or workflow based systems to provide process analysis. Process mining analyses historic data from single system logs to provide process information. While process mining is completely non-intrusive and hence cost-effective; it assumes that sequential events logs would be available from the information system, which is not always true. This approach may not be suitable for legacy systems where the process data is not stored in the format required by process mining algorithms and hence end to end process information may not be presented to the business user.

Business Provenance technology (Curbera, Doganata, Martens, Mukhi and Slominski 2008) helps trace end-to-end business operations across heterogeneous systems by collecting, co-relating and analyzing operational data. This approach involves creating a provenance graph for the specific application need based on which the provenance data is generated for extracting the required information about the business operations. The provenance data is generated by accessing application events from either event reporting middleware or by processing the application data and identifying the events. This approach can be used to gather, correlate and monitor process information for the specific need which is identified to create the provenance graph. The business process flow for each instance cannot be traced using such provenance graphs.

Our PIM approach and Business Provenance technology trace the process execution by identifying events of relevance from application data. The main difference is related to the non-intrusiveness of the approach. Business provenance uses specific instrumentation to extract the relevant information. This implies some intrusion into the existing systems. Instrumentation implies additional software routines and data which are to be added during compilation of the program, either by addition of routines to the program code and data or by linking routines from a suitable library. The provision of instrumentation adds extra code to the program which is not needed for the program's operation, thus increasing its size, and is hence not suitable for monitoring conventional operation of a process. Our approach uses application data created by the processes and hence does not affect the process execution in any manner. Some other differences are - we use business process model against which process instances are tracked, hence various rules and requirements for tracking can be modelled into the solution by the business user at a later stage. PIM uses the BPMN models to represent the process and provide analytics information in the same manner as BAM tools. Business provenance makes use of Open Provenance Method (Moreau, Freire, Futrelle, McGrath, Myers, and Paulson 2007) to create provenance graphs. Process analysis is done by querying the graphs, using graph patterns, etc. Since provenance graph is specific to the collected data and relationship among them, a different requirement for visibility (for e.g. compliance assurance instead of KPI adherence) would require a new provenance graph to be created.

CONCLUSIONS AND FURTHER WORK

PIM is a cost effective solution for monitoring end-to-end business processes. Its non-intrusiveness and ability to monitor processes across different systems is a key advantage. The process monitoring is done from persistent process execution data available in various data sources and adequately meets the monitoring requirements where users can take alternate action based on the alerts. While all business transaction data which is of any relevance is usually logged for further processing, a limitation of the current approach is that if a business activity does not leave a trace in persisted data, it cannot be tracked; manual activities and the calculations done in system memory fall within this category. But in reality there is no requirement to track all activities as seen in the above case. Identifying the problem areas and the activities to be tracked has to be done along with the business users and the IT group. Non-intrusiveness would also mean that it is a "near real-time" monitoring. The process execution is always one cycle ahead of the monitoring system, the monitoring lag is equal to the frequency of polling or reading of the process execution data. The process tracking will have this lag and hence the alerts will not be in perfect synchronization with the process execution. This drawback can be overcome by calculating the optimal polling frequency and scheduling some leading alerts. To handle any impact on performance, the monitoring can also be done off a data dump created at predefined frequencies instead of accessing the run-time database. PIM tool can also work in conjunction with BPM systems. If a part of the process is deployed on BPM software and rest of the process is on other systems, the process owner can still gain visibility over the complete process on a single dashboard.

The areas of subsequent research are on making the method more generic to monitor processes across different system architectures and on processes implemented as batch processes. Developing compliance related features and pointing out compliance violations is another planned area of research.

REFERENCES

1. Abadi D. J., Ahmad Y., Balazinska M. et al (2005)., The Design of the Borealis Stream Processing Engine, Second Biennial Conference on Innovative Data Systems Research (CIDR 2005),Asilomar, CA, January 2005.
2. Ammon R., Emmersberger C. and Springer F., Wolff C. (2008) Event-Driven Business Process Management and its Practical Application Taking the Example of DHL. 1st International workshop on Complex Event Processing for the Future Internet, Vienna, Austria, (28-30 September 2008).
3. Basili V. R. and Weiss D. M. (1984), "A methodology for collecting valid software engineering data", IEEE Transactions on Software Engineering, SE-10(6):728-738, November 1984.
4. Cohen M. A., Ramesh J. and Chen M., (2005) 'Reducing Business Surprises through Proactive, Real-Time Sensing and Alert Management', EESR '05: Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services.
5. Curbera, F., Doganata, Y., Martens, A., Mukhi, N. K., Slominski, A. (2008) Business Provenance – A Technology to Increase Traceability of End-to-End Operations. R. Meersman and Z. Tari (Eds.): OTM 2008, Part I, LNCS 5331, pp. 100–119.
6. Dumas M., van der Aalst W. M. P., and Ter Hofstede, (2005) "Process-aware information systems: Bridging people and software through process technology", John Wiley & Sons, Inc.
7. Grigori D., Casati F., Catellanos M., Dayal U., Sayal M. and Shan M. C., (2008) "Business Process Intelligence ", Computers in Industry, Volume 53, Issue 3, April 2004, Pages 321-343, Process / Workflow Mining, Elsivier.

8. Jourdan Z. , Rainer R. K. , Marshall T. E. (2008) Business Intelligence: An Analysis of the Literature, Information Systems Management, v.25 n.2, p.121-131, March 2008.
9. Kharbili M.E. (2008) Event-Based Decision Management in Compliance Management: A Discussion Paper. 1st International workshop on Complex Event Processing for the Future Internet, Vienna, Austria, (28-30 September 2008).
10. Leymann F., Reisig W., Thatte S.R., and van der Aalst W.M.P. (2006) The Role of Business Processes in Service Oriented Architectures, number 6291, Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.
11. Luckham D. (2002) "The Power of Events, An Introduction to Complex Event Processing in Distributed Enterprise Systems", Addison-Wesley Press, April 2002.
12. Luckham, D. (2004) The Beginnings of IT Insight: Business Activity Monitoring http://www.ebizq.net/topics/bam/features/4689.html, 2004, downloaded 2006-04-24.
13. McGregor C. and Schiefer J. (2004) A Web-Service based framework for analyzing and measuring business performance, Information Systems and E-Business Management, Volume 2, Issue 1, Apr 2004, pp. 89-110.
14. Moreau, L., Freire, J., Futrelle, J., McGrath, R., Myers, J., Paulson, P. (2007) The Open Provenance Model Technical Report, Provenance in Scientific Computing , http://eprints.ecs.soton.ac.uk/14979/ .
15. Sen S. (2008) Business Activity Monitoring Based on Action-Ready Dashboards And Response Loop. 1st International workshop on Complex Event Processing for the Future Internet, Vienna, Austria (28-30 September 2008).
16. van der Aalst, W., Weijters, A., Maruster, L. (2004): Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering 16(9),1128–1142.
17. Verner L. (2004), "The Challenge of Process Discovery", BP Trends; May 2004.
18. Wolf A. L., Rosenblum D. S. (1993). "A Study in Software Process Capture and Analysis", 2nd International Conference on the Software Process, Berlin, Germany, February 1993.